ML

2

04Ca #2
9-28-01
6-11-01

169.2065

PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	)	
	:	
GRAHAM STONEY	)	Examiner: Not Yet Assigned
	:	
	)	Group Art Unit: NYA
Application No.: 09/863,434	:	
	)	
Filed: May 24, 2001	:	
	)	
For: HIGHLY PIPELINED PRINTING SYSTEM ARCHITECTURE	:) :	June 15, 2001

Commissioner for Patents
Washington, D.C. 20231

CLAIM TO PRIORITY

Sir:

Applicant hereby claims priority under the International Convention and all rights to which he is entitled under 35 U.S.C. § 119 based upon the following Australian Priority Application:

PQ7723, filed on May 24, 2000.

A certified copy of the priority document is enclosed.

Applicant's undersigned attorney may be reached in our New York office by telephone at (212) 218-2100. All correspondence should continue to be directed to our address given below.

Respectfully submitted,

Attorney for Applicant

Registration No. 38,667

FITZPATRICK, CELLA, HARPER & SCINTO
30 Rockefeller Plaza
New York, New York 10112-3801
Facsimile: (212) 218-2200

NY_MAIN 177808 v 1

S.N. 09/863,434
Atty Dkt. No. 169.2065

AUSTRALIA

Patent Office
Canberra

I, JONNE YABSLEY, TEAM LEADER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. PQ 7723 for a patent by CANON KABUSHIKI KAISHA filed on 24 May 2000.

WITNESS my hand this
First day of June 2001

J R Yabsley

JONNE YABSLEY
TEAM LEADER EXAMINATION
SUPPORT AND SALES

CERTIFIED COPY OF
PRIORITY DOCUMENT

HIGHLY PIPELINED PRINTING SYSTEM ARCHITECTURE

Technical Field of the Invention

The present invention relates generally to printing systems, and in particular, to high performance printing systems. The present invention relates to a method and apparatus for pipeline data processing in printer systems.

Background Art

Conventional desktop and mainframe computer-based printing systems typically operate by processing pages in a document in a sequential manner. Accordingly, each page is individually processed, the individual processing for a particular page having to be completed before processing for a following page commences. As a consequence, the processing for a page does not begin until the previous page has been completely printed. The printing process itself typically consists of a number of sequential processing stages, each of which is made up of a combination of computer software and hardware based sub-processes. In the conventional printing model, each sub-process stage must be fully completed before the next sub-process stage can commence.

Many modern printer engines are capable of printing multiple pages during a single printing cycle. These engines must, however, be fed data continuously, in order to maintain the continuous printing cycle, and thus perform at their maximum rated engine speed and throughput. If a situation arises where raw data required for printing a particular page is not ready when required, the printer engine must cycle down, ie. go to a standby mode in an orderly fashion, and cease feeding input media, ie. paper, in order to prevent blank media appearing in the middle of the output document. Only when raw data is again available for the subsequent page, can the printer engine again begin processing.

This cycle down/cycle up process reduces engine throughput considerably, and of course, causes the printing system to operate below maximum performance levels. The presence of bottlenecks which are inherent in conventional sequential printing systems

makes it difficult to provide a continuous supply of raw print data to the printer engine as required. As printer engine speeds increase, this problem is exacerbated, since the speed with which the raw data must be supplied to the print engine also increases. Furthermore, as achievable print resolution increases, the volume of data required to process each print job also increases, typically at the square of the resolution increase, since the printed page is two-dimensional. The aforementioned technological advances place increasing strain on the conventional sequential processing model.

One present solution to the aforementioned problems is to "spool" job data at various stages in the print data flow processing. This approach can sometimes eliminate the cycle down/cycle up problem if applied to a relatively short print run. A penalty is incurred, however, since spooling adds additional overhead to the print data processing,. because data must be typically copied to, and from, a relatively slow storage device such as a hard disc. Furthermore, spooling introduces an arbitrary restriction on the length of a print job, since data for an entire job must typically be stored in spooling systems. Furthermore, the spooling approach also introduces additional start-up delay, before the first page of a job can be printed.

Disclosure of the Invention

It is an object of the present invention to substantially overcome, or at least ameliorate, one or more disadvantages of existing arrangements.

According to a first aspect of the invention, there is provided a method of data processing for a printing system which comprises a sequence of pipeline processes, said method comprising, for a current pipeline process, steps of:

reading input data from an upstream pipeline process;

operating upon said input data if an internal buffer of said current pipeline process is not full;

stalling said upstream pipeline process, if said internal buffer is full; and

writing said input data, having operated thereupon, to a downstream pipeline process, if said downstream pipeline process is not stalling said current pipeline process.

According to another aspect of the invention, there is provided an apparatus for implementing any one of the aforementioned methods.

Brief Description of the Drawings

A number of preferred embodiments of the present invention will now be described with reference to the drawings, in which:

Fig. 1 shows a block diagram representation of hardware components in a printing system according to a preferred embodiment of the invention;

Fig. 2 shows a pipeline arrangement for the host computer of Fig. 1;

Fig. 3 shows pipeline processing details for the printer controller of Fig. 1;

Fig. 4 shows pipeline details for the print engine of Fig. 1;

Fig. 5 is a schematic block diagram of a general purpose computer upon which an embodiment of the present invention can be practised;

Fig. 6 is a flowchart of method steps relating to processing of data from an upstream device; and

Fig. 7 is a flowchart of method steps relating to processing of data for a downstream device.

Detailed Description including Best Mode

Where reference is made in any one or more of the accompanying drawings to steps and/or features, which have the same reference numerals, those steps and/or features have for the purposes of this description, the same function(s) or operation(s), unless the contrary intention appears.

In the context of this specification, the word "comprising" means "including principally but not necessarily solely" or "having" or "including" and not "consisting only of". Variations of the word comprising, such as "comprise" and "comprises" have corresponding meanings.

In the context of this specification, the term "line" is used to donate a data connection between process blocks, and/or between sub-process blocks. Data

connections can adopt a variety of physical forms including hard-wired and wired connections, and can also represent software calls between software processes.

Fig. 1 shows major hardware components of a printing system, according to a preferred embodiment of the invention. A host computer 100 is provided with a host interface 102, to which a printer controller 104 is connected. The printer controller 104 is, in turn, connected by means of an engine interface 106 to a print engine 108. The print engine 108 is responsible for feeding and marking print data on the required output media, this media typically being paper. Typically, the printer controller 104 and the print engine 108 are regarded as a single entity by an end user, however they can typically be sourced from different Original Equipment Manufacturers (OEMs).

Fig. 2 illustrates pipeline processes operating within the host computer 100 in a preferred embodiment of the invention. The figure elaborates on the various data processing stages within major hardware and software components. Each processing stage operates in parallel, ie. concurrently, a current stage accepting data from a previous stage, and sending data to a downstream stage as output data from the current stage becomes available. Each stage also provides a "stall control" signal back to the previous, ie. upstream, stage, in order to indicate when data flow should be quenched to avoid causing overflow of internal buffers in the downstream pipeline.

Accordingly, a host application 200 makes printer Graphical Device Interface (GDI) calls on a line 202, these calls describing the page image to a printer driver 204. The printer driver 204 in turn generates a page description on a line 206, this description being produced in any applicable page description language accepted by the printer. Page description languages include Postscript, PCL, LIPS, or alternatively, printer specific formats which can be proprietary. This page description is written to a spooler 208 of the Windows printing system. The spooler 208 in the preferred embodiment writes the data on a line 210 directly to a printer port monitor 212, without spooling the data onto a disc. The port monitor 212 in turn writes the data on a line 214 to the applicable host hardware interface 216, which can be any interface capable of communication with the printer. Examples of such interfaces include IEEE-1284, TCP/IP over Ethernet, or any other

suitable printer interface. Job data from the hardware interface 216 is sent on a line 218 to a printer controller receiving hardware interface 300 (see Fig. 3).

Fig. 3 shows pipeline processing details for the printer controller 104 of Fig. 1. The receiving hardware interface 300 sends the aforementioned job data on a line 302 to a page description language interpreter 304, which interprets the contents of the job data as it arrives, thereby generating a display list on a line 306 which is written to a renderer software interface 308. The display list on the line 306 is thereby transferred on a line 310, into a display list memory 324. Thereafter, the contents of the display list memory 324 are transferred on a line 328, possibly using direct memory access (DMA), to a rendering hardware process 312, which generates raw pixel data on a line 314. The raw pixel data on the line 314 can utilise the RGB colour space, and is sent to the colour converter 400 of the print engine 108 (see Fig. 4).

Fig. 4 shows pipeline process details inside the print engine 108. The incoming raw pixel data on the line 314 is input into a colour converter 400, which converts the data to a CMYK space, or alternatively, to whichever colour space is required by a subsequent marking process 408. The colour converter 400 outputs the CMYK (or alternative) pixel data on a line 402 into an output processing sub-process 404, which generates host processed marker data on a line 406 as required by the marking engine 408. The marking engine 408 then proceeds to generate the final image on the appropriate output medium.

The above description has presented a forward data flow path from the host application 200 and through to the marking engine 408. It is noted, however, that each stage in the aforementioned cascade of sub-processes typically has some internal buffering. Each stage can, in addition, signal a "stall condition" back to an upstream stage in the pipeline, should this internal buffer become full.

It is noted that the processing stages can be implemented using a mix of hardware and/or software. The "signalling" between stages will, accordingly, be either a hardware signal on a connecting line (for example between two hardware stages), or alternatively the "signal" takes the form of a software call, or indication, (eg. between two

software stages). This applies both to signals on the forward path, and on the reverse path which relates to signalling of the stall conditions.

Stall signals flow in a reverse direction, and are depicted as dashed arrows in the figures, indicating that these signals are only generated if necessary to prevent buffer overflow. The forward data flow is depicted by forward arrows using solid lines, indicating that this data is expected to flow in the normal course of printer operation. Stall condition signalling prevents data overflow at any stage, while allowing maximum concurrent processing to proceed.

The stall control, considered from the marking engine 408 end of the series of cascaded sub-processes, operates as follows. The marking engine 408 accepts data synchronously from the output processing step 404 on the line 406 as described. When the marking engine 408 is busy, it signals this "busy" state via a hardware signal on a dashed line 410 to the output processing step 404 which in turn stalls the colour converter 400 via a hardware signal on a dashed line 412.

The colour converter 400 in turn stalls the rendering hardware sub-process 312 by means of a signal on a dashed line 316. The rendering hardware 312, then stalls the display list memory 324, by means of a signal on a dashed line 326, this being performed if the rendering hardware 312 local memory 324 becomes full. When the display list memory 324 fills, it stalls the renderer software interface 308 by means of a signal on a dashed line 318. This stall signal on the line 318 propagates further in the reverse direction to the interpreter sub-process 304 on a line 320, blocking further write operations to the renderer 308. In turn, the interpreter 304 ceases reading input date from the receiving hardware interface 300, this being achieved by means of a stall signal on a dashed line 322. The receiving hardware interface 300, in turn, stalls the host hardware interface 216 by means of a signal on a dashed line 220, causing the host hardware interface 216 to block write calls from the port monitor by means of a stall signal on the a dashed line 222. The port monitor, in turn, blocks write calls from the spooler by means of a signal on a dashed line 224, this in turn being propagated to block write signals from the printer driver 204 by means of a stall signal on a dashed line 226. The printer driver,

in turn, prevents the GDI calls from the host application 200 by means of a stall signal on a dashed line 228.

It should be noted that each of the aforementioned stall signals is asserted when a relevant downstream module cannot accept any further data, typically because an internal buffer associated with that downstream module is full. In this manner, a stall signal generated by the marking engine 408 can propagate all the way back to the host application 200. It should, however, be noted that stall signals can originate anywhere along the series of pipelined sub-processes, and indeed can be generated at more than one location substantially simultaneously.

The disclosed method of data processing for printer systems can be implemented in dedicated hardware elements, and/or dedicated software elements, or a mix of the aforementioned process types. Having regard to the particular implementation depicted in Figs. 2 to 4, processing stages 200, 204, 208 and 212 are implemented in software. Accordingly, the interposed communication "connections", eg. 202 and 228, take the form of software calls, or indications. Processing stages 216 and 218, ie the interface processes, are implemented in a mixture of software and hardware. Accordingly, the connections 218 and 220 can take the form of a hardware/software mix, depending on the specifics of the interfaces used (eg. IEEE 1284 Ethernet, and so on). Processors 304 and 308 are implemented in software, while processes 324 and 312 are implemented in hardware. The connections between these various stages carry either hardware based signals, or software calls, in accordance with the specifics of the aforementioned implementation.

The method of data processing for a printing system is preferably implemented in dedicated hardware such as one or more integrated circuits performing the functions or sub functions of data processing for a printing system. Such dedicated hardware may include graphic processors, digital signal processors, or one or more microprocessors and associated memories.

The method of data processing for a printing system can also be practiced using a conventional general-purpose computer system 500, such as that shown in Fig. 5

wherein the processes of Figs. 6 and 7 may be implemented as software, such as an application program executing within the computer system 500. In particular, the steps of data processing for a printing system are effected by instructions in the software that are carried out by the computer. The software may be divided into two separate parts; one part for carrying out the data processing for a printing system methods, and another part to manage the user interface between the latter and the user. The software may be stored in a computer readable medium, including the storage devices described below, for example. The software is loaded into the computer from the computer readable medium, and then executed by the computer. A computer readable medium having such software or computer program recorded on it is a computer program product. The use of the computer program product in the computer effects an apparatus for data processing for a printing system in accordance with the embodiments of the invention.

The computer system 500 comprises a computer module 501, input devices such as a keyboard 502 and mouse 503, output devices including a printer 515 and a display device 514. A Modulator-Demodulator (Modem) transceiver device 516 is used by the computer module 501 for communicating to and from a communications network 520, for example connectable via a telephone line 521 or other functional medium. The modem 516 can be used to obtain access to the Internet, and other network systems, such as a Local Area Network (LAN) or a Wide Area Network (WAN).

The computer module 501 typically includes at least one processor unit 505, a memory unit 506, for example formed from semiconductor random access memory (RAM) and read only memory (ROM), input/output (I/O) interfaces including a video interface 507, and an I/O interface 513 for the keyboard 502 and mouse 503 and optionally a joystick (not illustrated), and an interface 508 for the modem 516. A storage device 509 is provided and typically includes a hard disk drive 510 and a floppy disk drive 511. A magnetic tape drive (not illustrated) may also be used. A CD-ROM drive 512 is typically provided as a non-volatile source of data. The components 505 to 513 of the computer module 501, typically communicate via an interconnected bus 504 and in a manner which results in a conventional mode of operation of the computer

system 500 known to those in the relevant art. Examples of computers on which the embodiments can be practised include IBM-PC's and compatibles, Sun Sparcstations or alike computer systems evolved therefrom.

Typically, the application program of the embodiment is resident on the hard disk drive 510 and read and controlled in its execution by the processor 505. Intermediate storage of the program and any data fetched from the network 520 may be accomplished using the semiconductor memory 506, possibly in concert with the hard disk drive 510. In some instances, the application program may be supplied to the user encoded on a CD-ROM or floppy disk and read via the corresponding drive 512 or 511, or alternatively may be read by the user from the network 520 via the modem device 516. Still further, the software can also be loaded into the computer system 500 from other computer readable medium including magnetic tape, a ROM or integrated circuit, a magneto-optical disk, a radio or infra-red transmission channel between the computer module 501 and another device, a computer readable card such as a PCMCIA card, and the Internet and Intranets including email transmissions and information recorded on websites and the like. The foregoing is merely exemplary of relevant computer readable mediums. Other computer readable mediums may be practiced without departing from the scope and spirit of the invention.

Fig. 6 shows a flowchart of method steps for a port process 610 which "runs" on an upstream input port (eg 230, see Fig. 2) of a representative pipeline process (eg 204) described in Figs. 2 to 4. On a point of terminology, the "port process" 610 is to be differentiated from the "pipeline process" 204 (ie the printer driver 204) on whose upstream input port 230 the port process 610 runs. In a step 600, the port process 610 waits for input data from an upstream pipeline process. Thereafter, in a testing step 602, an internal buffer of the pipeline process 204 on whose upstream port the port process 610 is running is checked for overflow. If an overflow condition is detected, then the port process 610 is directed in accordance with a "yes" arrow to a stalling step 604, which stalls the upstream device until buffer space in the pipeline process 204 becomes available. Once such space becomes available, the port process 610 is directed to a step

606 which proceeds to read input data from the upstream device into the internal buffer of the pipeline process 204. Returning to the testing step 602, if the internal buffer of the pipeline process 204 is not experiencing an overflow condition, then the port process 610 is directed in accordance with a "no" arrow to the step 606. After the step 606 reads input data from the upstream device into the internal buffer of the pipeline process 204, the port process 610 is directed in accordance with an arrow 608 back to the step 600, where the port process 610 waits for further input data from the upstream process.

Fig. 7 shows a flowchart of method steps describing a port process 710 which runs at the output (ie. downstream facing) port (eg 232, see Fig. 2) of each pipeline process (eg the host hardware interface pipeline process 216) described in Figs. 2 to 4. In a step 700, the port process 710 waits for data to appear in an internal buffer of the pipeline process 216. Once such data is detected, a testing step 702 considers whether the output of the pipeline process 216 is being stalled. If this is the case, then the port process 710 is directed in accordance with a "yes" arrow to a wait step 704, which freezes operation of the pipeline process 216 until the downstream device removes the present stall condition. Thereafter, the port process 710 is directed to a write step 706, which writes data from the internal buffer of the pipeline process 216 to the output port thereof. Returning to the testing step 702, if the output is not being stalled (ie. by the downstream device), then the process 710 is directed in accordance with a "no" arrow to the step 706. After step 706 the process 710 is directed in accordance with an arrow 708 back to the wait step 700.

Each of the pipeline processes shown in Figs. 2 to 4 (from the printer driver 204 to the output processing device 404) runs an input port process 610 and an output port process 710 on the input and output ports respectively of the pipelined process.

Industrial Applicability

It is apparent from the above that the embodiment(s) of the invention are applicable to the printing industry, and indeed to any industry in which high performance printing processes are used or required.

The foregoing describes only one embodiment of the present invention, and modifications and/or changes can be made thereto without departing from the scope and spirit of the invention, the embodiment being illustrative and not restrictive.

Claims:

1. A method of data processing for a printing system which comprises a sequence of pipeline processes, said method comprising, for a current pipeline process, steps of:

reading input data from an upstream pipeline process;

operating upon said input data if an internal buffer of said current pipeline process is not full;

stalling said upstream pipeline process, if said internal buffer is full; and

writing said input data, having operated thereupon, to a downstream pipeline process, if said downstream pipeline process is not stalling said current pipeline process.

2. A method according to claim 1, wherein said operating step comprises at least one of:

processing said input data; and

storing said input data.

3. A method according to claim 1, wherein said sequence of pipeline processes comprises at least one of hardware processes and software processes.

4. A method according to claim 1, wherein said operating step associated with each process in said sequence of pipeline processes is performed substantially concurrently, by all processes in said sequence.

5. A printing system implementing the aforementioned method.

6. A computer program product including a computer readable medium having recorded thereon a computer program for performing data processing for a printing system as described in the aforementioned method.

7. A method of data processing for a printing systems substantially as described herein, with reference to the accompanying drawings.

8. A printer system substantially as described herein, with reference to the accompanying drawings.

9. A computer program product including a computer readable medium having recorded thereon a computer program substantially as described herein, with reference to the accompanying drawings.

DATED this Twenty-third Day of May, 2000

Canon Kabushiki Kaisha

Patent Attorneys for the Applicant

SPRUSON & FERGUSON

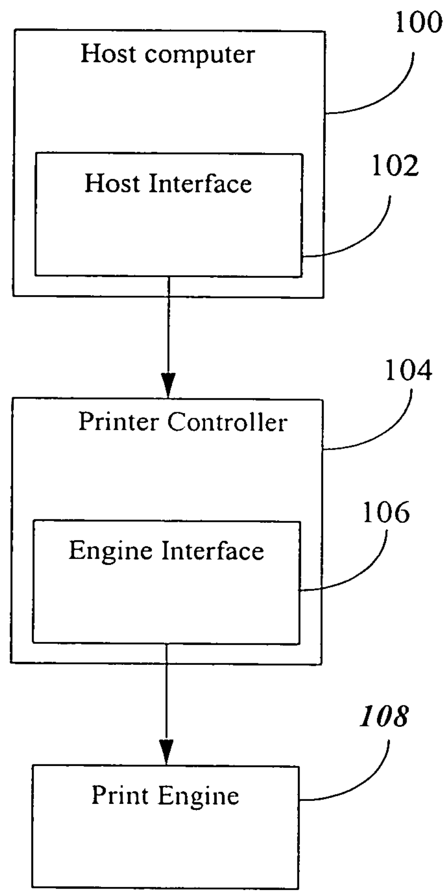
100
Host computer
102
Host Interface
104
Printer Controller
106
Engine Interface
108
Print Engine

Fig. 1

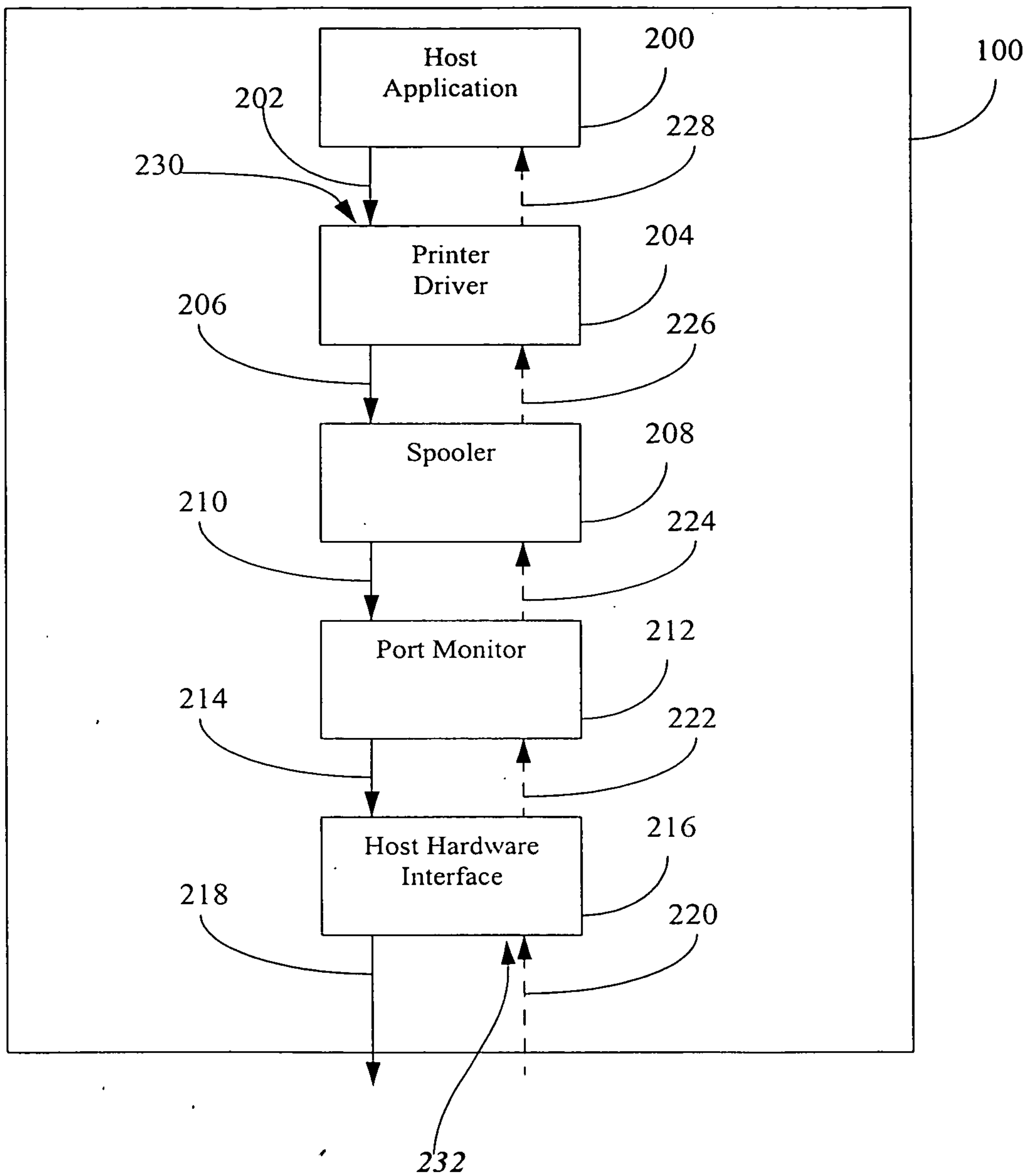
Host
Application
200
100
202
228
230
Printer
Driver
204
206
226
Spooler
208
210
224
Port Monitor
212
214
222
Host Hardware
Interface
216
218
220
232

Fig. 2

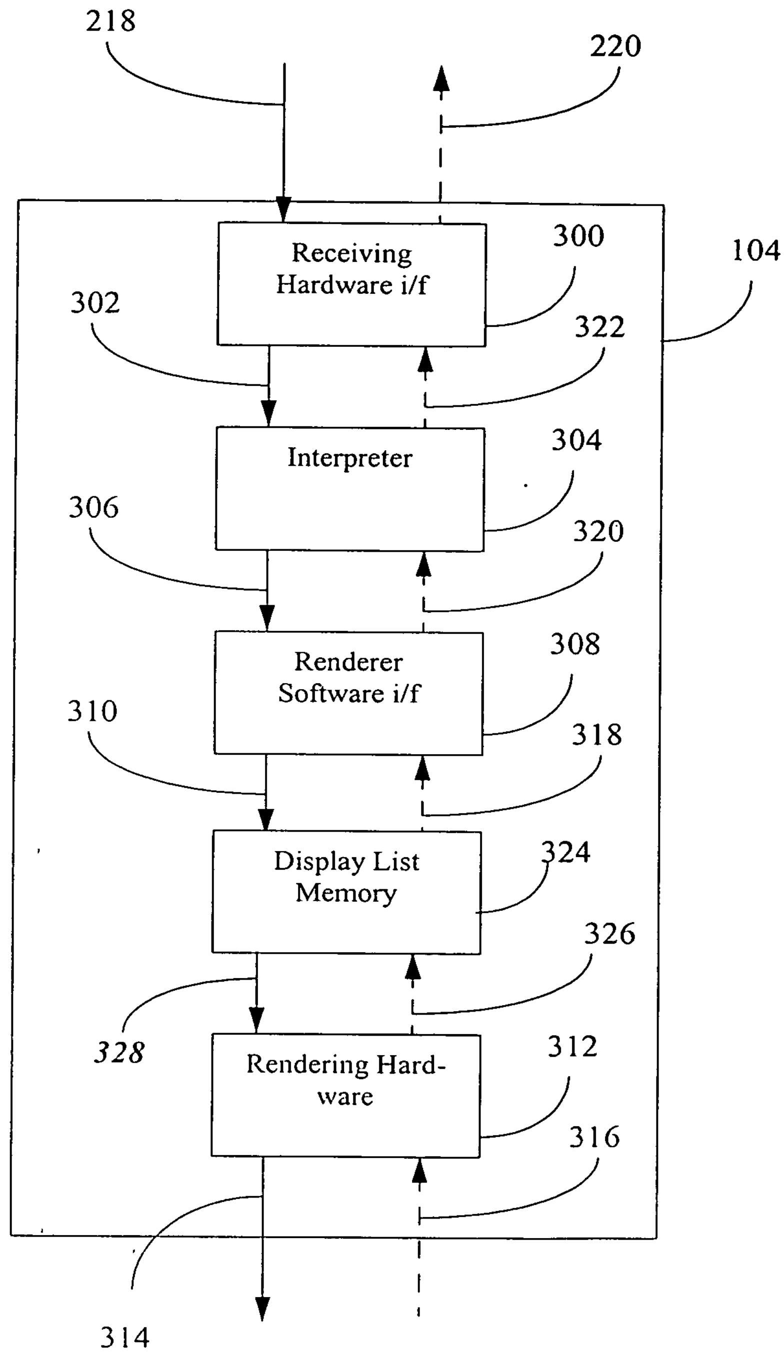
218
220
Receiving Hardware i/f
300
104
302
322
Interpreter
304
306
320
Renderer Software i/f
308
310
318
Display List Memory
324
326
328
Rendering Hard-ware
312
316
314

Fig. 3

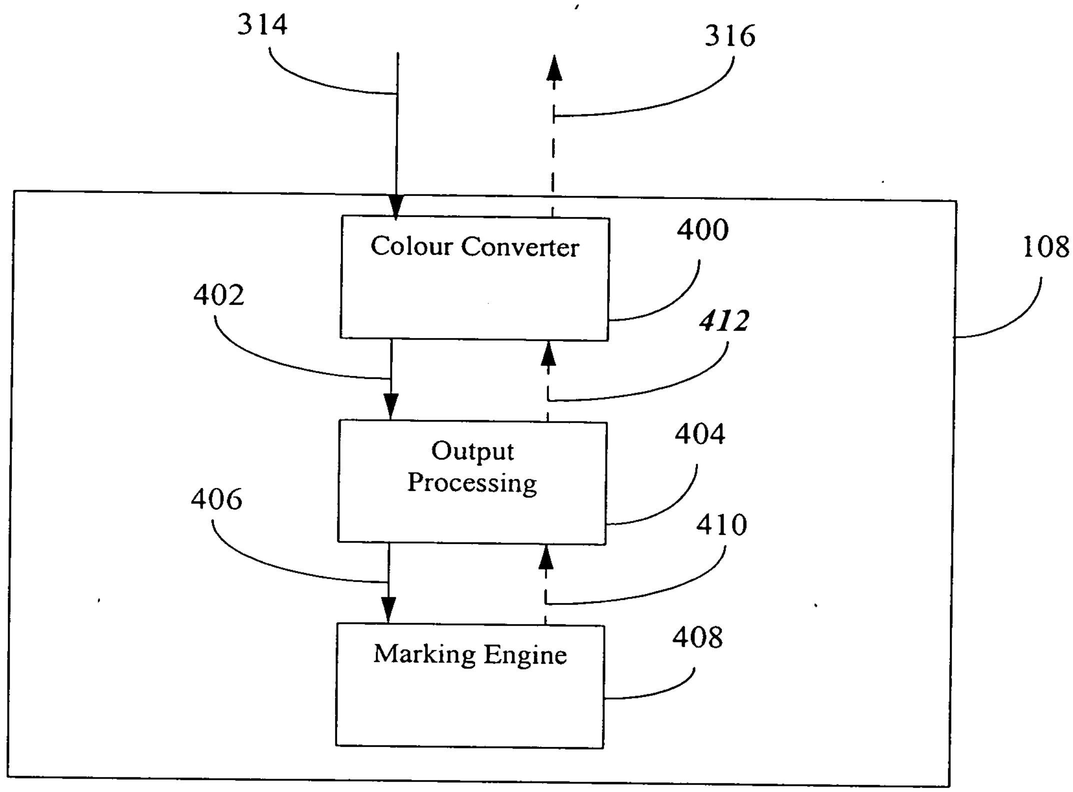
314
316
Colour Converter
400
108
402
412
Output Processing
404
406
410
Marking Engine
408

Fig. 4

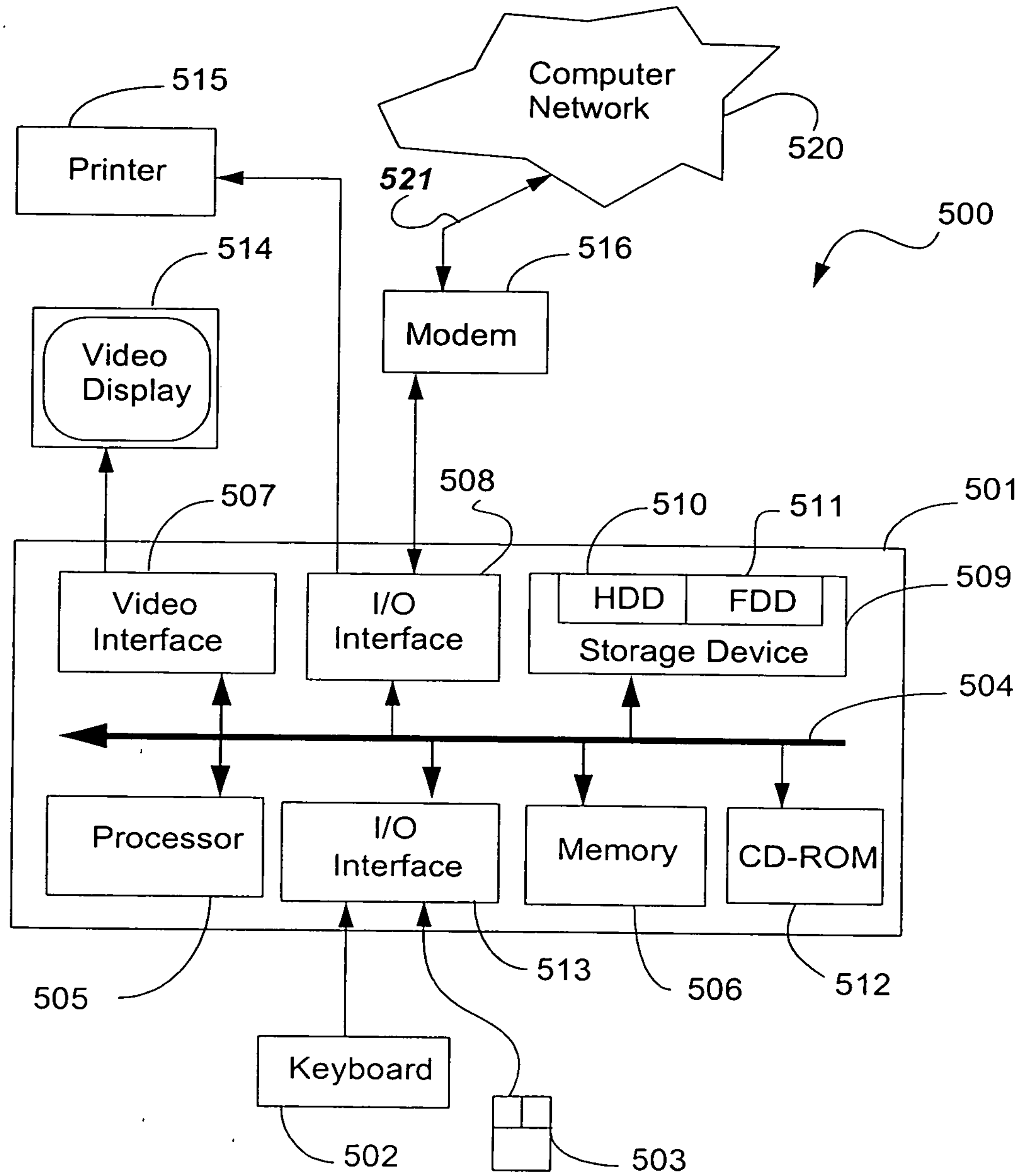
515
Printer
Computer
Network
520
521
500
514
516
Video
Display
Modem
507
508
510
511
501
509
Video
Interface
I/O
Interface
HDD
FDD
Storage Device
504
Processor
I/O
Interface
Memory
CD-ROM
505
513
506
512
Keyboard
502
503

Fig. 5

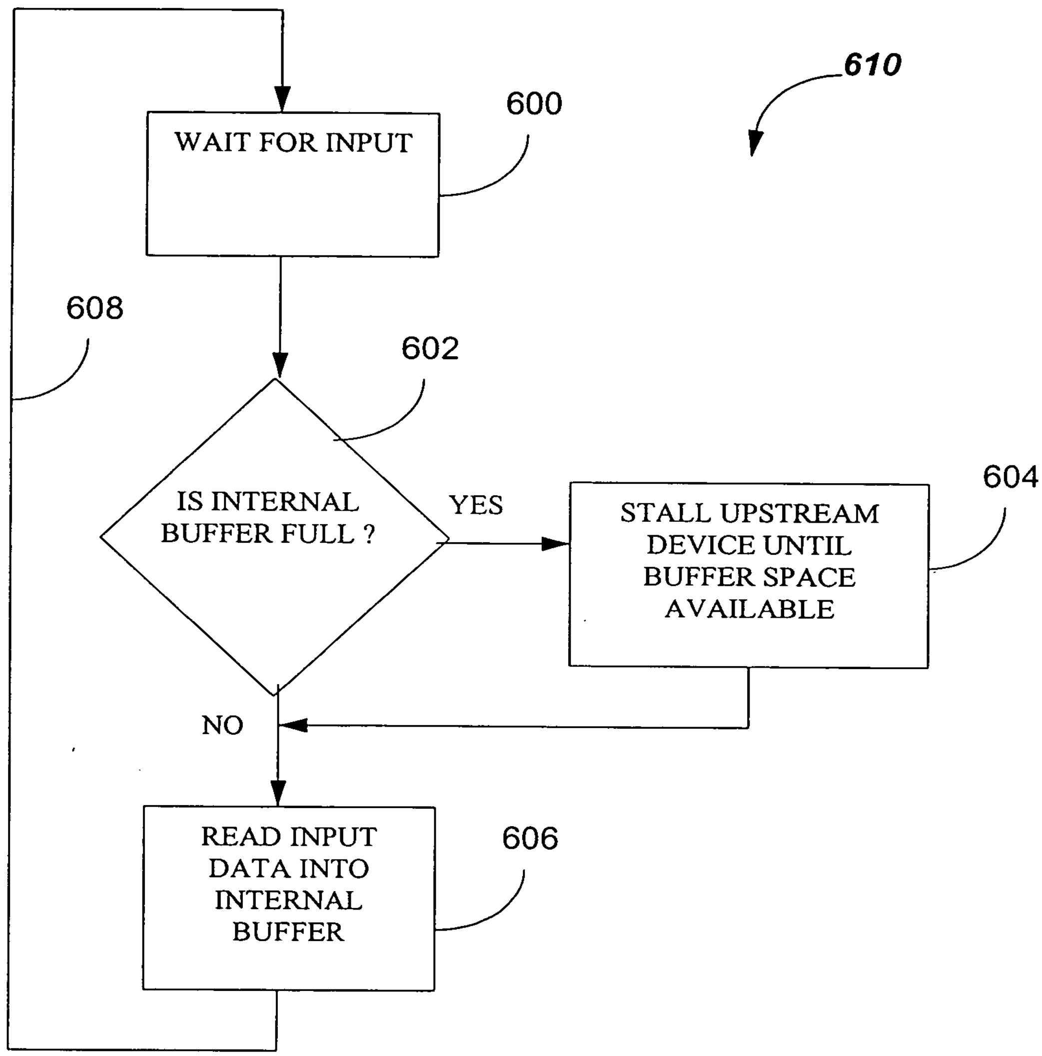
610
600
WAIT FOR INPUT
608
602
IS INTERNAL BUFFER FULL ?
YES
604
STALL UPSTREAM DEVICE UNTIL BUFFER SPACE AVAILABLE
NO
READ INPUT DATA INTO INTERNAL BUFFER
606

Fig. 6

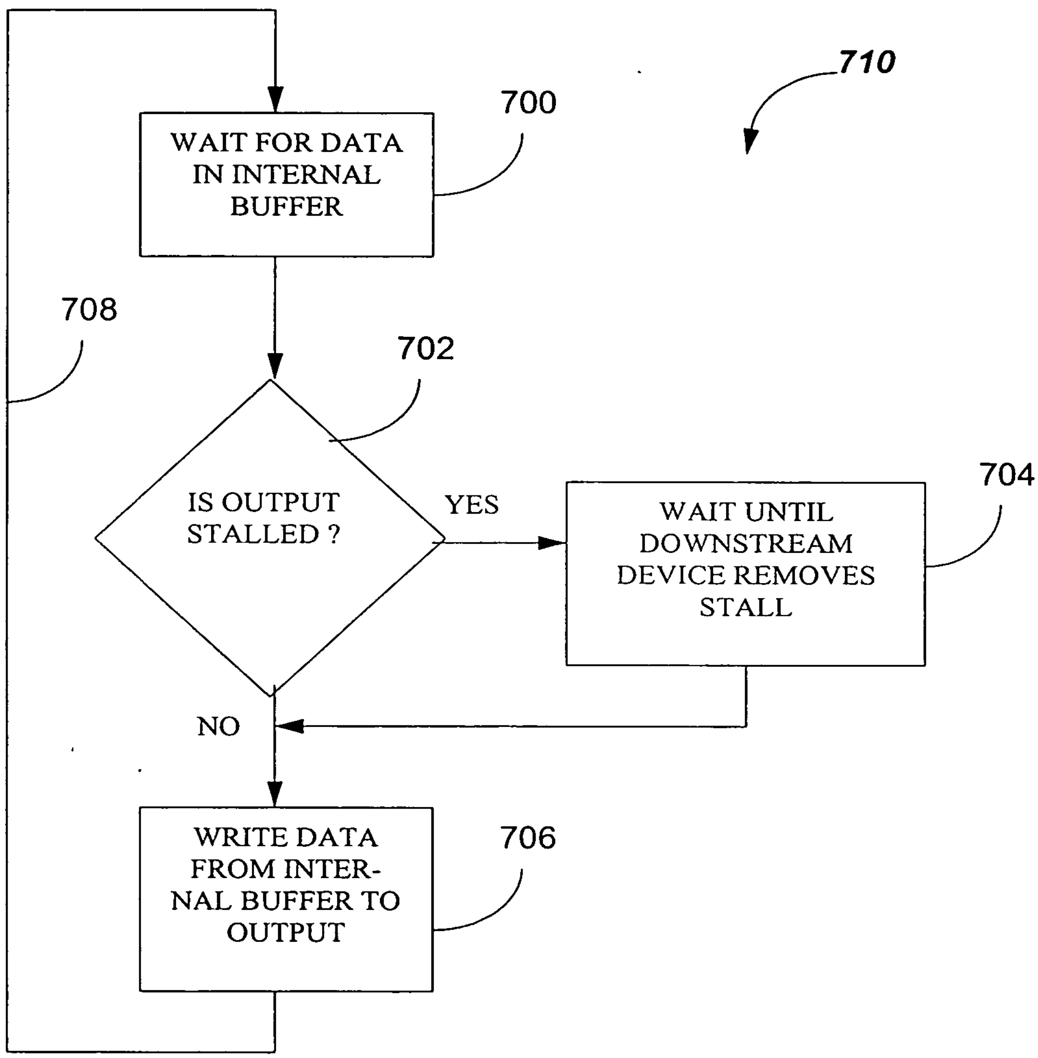
710
700
WAIT FOR DATA IN INTERNAL BUFFER
708
702
IS OUTPUT STALLED ?
YES
704
WAIT UNTIL DOWNSTREAM DEVICE REMOVES STALL
NO
706
WRITE DATA FROM INTER-NAL BUFFER TO OUTPUT

Fig. 7